18th Brazilian Congress of Thermal Sciences and Engineering
November 16–20, 2020 (Online)

ENC-2020-0704 Automatic Structured Mesh Generation Around Two-dimensional Airfoil Applied to Aerodynamics Computational

Pedro Henrique de Araújo Bitencourt
Henrique Matos Campos
Aluisio Viais Pantaleão
São Paulo State University (UNESP), School of Engineering Ilha Solteira. Avenida Brasil Sul, 56 Centro, Ilha Solteira-SP, 15385-000
araujo.bitencourt@unesp.br, henrique.campos@unesp.br, aluisio.pantaleao@unesp.br

Abstract. *The use of numerical methods to forecast aerodynamic phenomena have an important role in the industry, and the generation of meshes is an extremely important step for the expected results. This work seeks to develop the automatic generation of structured meshes through integration between the Python language and the free software OpenFoam, which interprets, through the blockMesh tool, a dictionary, then generate the structured mesh. The comparison with unstructured meshes was used in order to verify the computational efficiency and accuracy of the generated structured meshes.*

Keywords: *CFD, aerodynamics, structured mesh, OpenFoam, Python*

1. INTRODUCTION

Engineering has evolved in recent years trying to reduce the amount of experiments in the industry, considering the large economic expenses. One of the solutions to this problem is the numerical simulations, since with one computational structure, it is possible to obtain results equivalent to several experiments, thus, reducing costs. Therefore, several researches and evaluations are regarding numerical simulations. They use computational meshes for their applications, and the meshing generation method varies a lot according to the degree of geometric complexity, since the more complex, the greater the difficulty in developing the computational mesh. The literature reports several generation methods, including structured, unstructured and mixed meshes. The choice of a method is due to the search for the best cost-benefit of generation time linked to greater efficiency in the convergence rate and accuracy obtained in the solution of the problem.

The methods to automatic generate meshes are being studied and developed by countless researchers. Within the aspect of unstructured meshes (Pantaleão *et al.*, 2003) developed an automatic generator for an unstructured mesh of triangular elements with three vertices, in which it was produced by a computer code using the Delaunay algorithm applied to the element method finite. (Benoit and Péron, 2012) researched a method to automatically generate structured meshes around two-dimensional bodies for CFD simulations, with a method that consists of avoiding low quality meshes for locally pointed bodies, such as the trailing edge of an airfoil and is able to automatically perform flow simulation around any geometry made of poly-lines. Also (Schmidt *et al.*, 2012) developed a new library for the software OpenFOAM that can be used for the automated generation of structured meshes, the library provides tools for the organization of a large number of blocks and is used prior to OpenFOAM's native block mesher blockMesh.

In the paper of (Lu *et al.*, 2018), the authors present a method of automatic generation of structured meshes in arbitrary aircraft, together with a method of automatic generation of a boundary layer mesh using multi-blocks structured from a superficial mesh, but there was no automatic generation of the textit far field. (Liu and Hu, 2018) studied the fluid-structure interaction problems based on his own previous research on the method of refining structured mesh adaptable to in-compressible flows. (Zhang and Jia, 2018) developed an algorithm for the automatic generation of structured meshes applicable to two-dimensional domains with complex geometries.

In the work of (Yu *et al.*, 2019), the algorithm on automatic mesh generation was further studied, which was more robust on the defective CAD geometries comparing com previous cases involving hydrogen, three cases on a leaking hydrogen tank, a steam generator and a single car garage were studied with regards to the validation of the mesh generation approach. (Faghih-Naini *et al.*, 2020) proposed a method implemented within the code generation framework of the ExaStencils project using the SymPy Python library, in which the new formulation uses block-structured triangular meshes automatically generated for a given number of blocks.

(Beaufort *et al.*, 2020) used a robust pipeline to handle triangulations, based on the computation of a one-to-one parametrization for automatically selected patches of input triangles, which made each patch easier to remeshing and was implemented in the open source mesh generator Gmsh. And (Lu *et al.*, 2020) implemented, from his previous work, a interactive structured mesh generation software called NNW-GridStar, with accelerated techniques, which consist of automatic boundary-layer mesh generation, rapid block assembly, multiblock stretching, and O-type block stretching.

This research developed a method to automatically generate structured meshes, including the far field, with the outline of 2D lifting surfaces, using free software, the Python language and OpenFoam, this approach leads to a high cost benefit, since it has a fast capacity of generation, the mesh has high quality and the access to the softwares are easy, with no costs. Also this research helps the University research concerning hydrodynamics profiles and have great interest from the aeronautic and energy(turbine generated) industries. In order to compare the performance of the mesh with the literature, the airfoils NACA 0012 and NACA 0021 were used.

2. METHODOLOGY

The methodology of the project consists in the integration between the selected simulation software and the programming language, in this case the OpenFoam software and the Python language were chosen, since in addition to being open source, there is a tool for generating structured meshes within the software.

Given the points of the desired airfoil, using the Python programming language, the code calculates the far field, and generates the parameters necessary to construct a mesh, so the file is automatically generated. In this way, the blockMesh tool of OpenFoam (2020) was used, in which it interprets the dictionary generated by the Python language, and thus relates the data of points, faces, cells and mesh divisions in a new dictionary.

In most applications, blockMesh is used as a pre-mesher for OpenFOAM's snappyHexMesh, which automatically generates a mesh around arbitrary surfaces, starting from the structured block mesh, but since we will only generate structured meshes, the blockMesh tool is indicated.

The blockMesh tool generates a strutured mesh, that is explained in Moukalled *et al.* (2015). Which inform that in this kind of mesh, every cell in the interior of the domain is connected to the same number of neighboring cells, which can be identified using the indices, like i,j,k, normally used in the x,y,z coordinates, and that it can be directly accessed by incrementing or decrementing the respective indices.

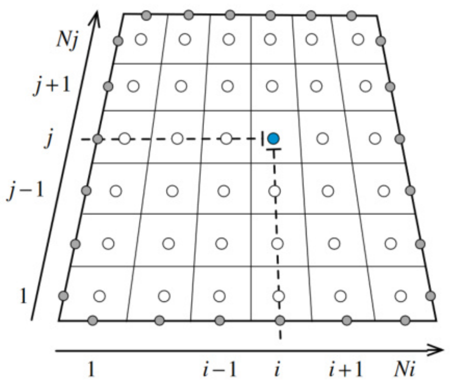

Figure 1. Example concerning the topology of a structured mesh, Moukalled *et al.* (2015).

Also, the indices system integrates with the equations, because generally global indices are used when building all the equations of the entire system over the computational domain, while local indices are usually applied to define the local template for an element. In the case of structured grid systems, the local indices can be interchangeable with global indices.

The generation of structured mesh by the blockMesh tool is through a dictionary named "blockMeshDict" in which it is located in the folder "system" of the case under analysis. Basically, as explained,it decompose the domain geometry into several three-dimensional hexahedron blocks. The vertices allow the generation of lines, arcs and splines, in addition to specifying the number of cells in each direction of the block, the whole set being characterized as a mesh.

The method utilized is simplified in the diagram shown in Figure 2, in which the gray blocks correspond to the code reported here.

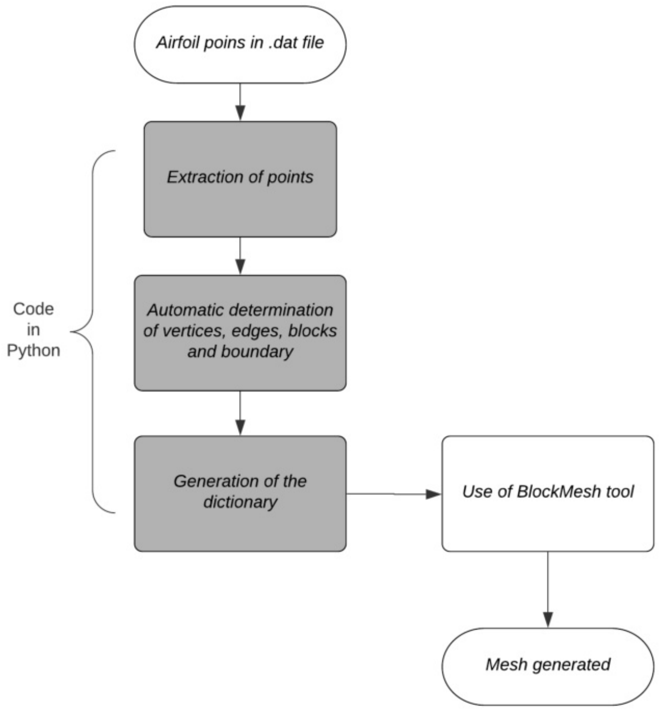
Airfoil poins in .dat file
Extraction of points
Automatic determination of vertices, edges, blocks and boundary
Generation of the dictionary
Use of BlockMesh tool
Mesh generated
Code in Python

Figure 2. Diagram of the method used to generate the mesh.

2.1 Analyzed test cases

To verify the quality of the meshes generated by the Python code, we will test different flow conditions.

The first test case verified was found in Rumsey (2019) and consists of an incompressible, turbulent, and steady flow over a NACA 0012 airfoil. Table 1 presents more details about the flow conditions for the test case.

The second test case defined in the study was for the airfoil NACA 0021, Wolfe and Ochs (1997) presents more details about the test case. This test case also consists of an incompressible, turbulent, and steady flow. Table 1 presents more information about the flow conditions for the test case.

Table 1. Airfoil validation cases information.

Test case	**NACA 0012**	**NACA 0021**
Reference	Rumsey (2019)	Wolfe and Ochs (1997)
Prandtl Number	0,72	0,72
Temperature [K]	299,85	299,85
Mach Number	0,15	0,20
Chord [m]	1	1
Reynolds Number	$6{\cdot}10^6$	$1{,}5{\cdot}10^6$
Chord [m]	1	1
Angle of Atack [°]	0	0

Since all the analyzed test cases use air as fluid, it was considered as an ideal gas. With this adoption, to evaluate the dynamic viscosity, we used Sutherland's Law.

As all the flows analyzed are turbulent, to solve the closure problem, was adopted the Spalart and Allmaras turbulence model.

2.2 Numerical Simulation

For the numerical simulation was used OpenFOAM v7 (OpenFOAM Foundation (2020) presents more information about the software). Since all the available test cases are incompressible, turbulent, and steady flow problems, was used simpleFoam as the numerical solver (a pressure-velocity coupling solver based on the SIMPLE algorithm).

For the numerical analysis, Gauss linear was the gradient schemes employed. Gauss linearUpwind was the divergence scheme used for velocity and turbulent viscosity. Gauss linear corrected was used as the laplacian schemes. As interpolation schemes were used linear interpolation profile.

The solvers adopted were GAMG using GaussSeidel smoother for the pressure and smoothSolver using GaussSeidel smoother for all the other properties.

The boundary conditions adopted were:

- Farfield boundary conditions (freestream Velocity in all farfield regions);
- Viscous wall in profile studied (noSlip boundary condition).

3. RESULTS

3.1 Analisys of NACA 0012 airfoil

The results of the mesh generation and simulation of a NACA 0012 are disposed below.

3.1.1 Mesh

Using the points of a NACA 0012 and the method explained in the methodology, the far field has a size 20 times bigger than the airfoil chord. The mesh disposed in the following images (Figures 3 and 4) was generated for an $y+$ value of 60, with approximately 2 millions hexahedral elements, since OpenFoam operates with a three dimensional mesh, even for bi-dimensional cases. Other two mesh with different values of $y+$ were generated for CGI analysis.

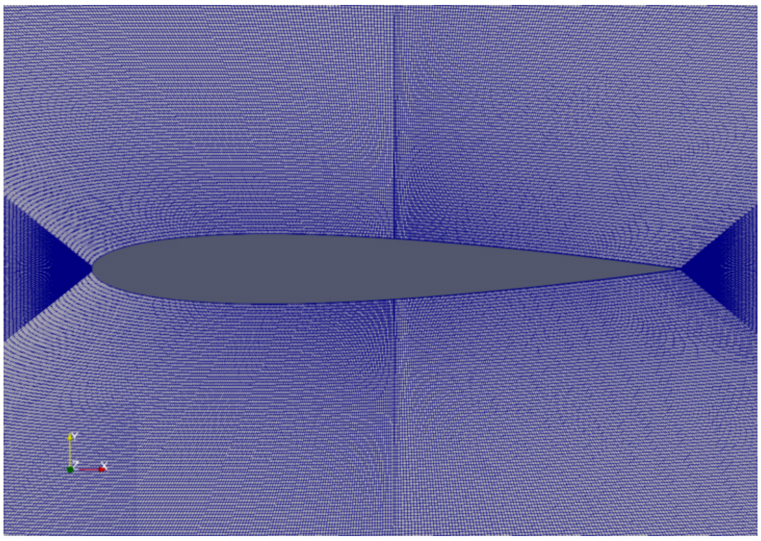
Figure 3. Demonstration of the mesh generated by Authors.

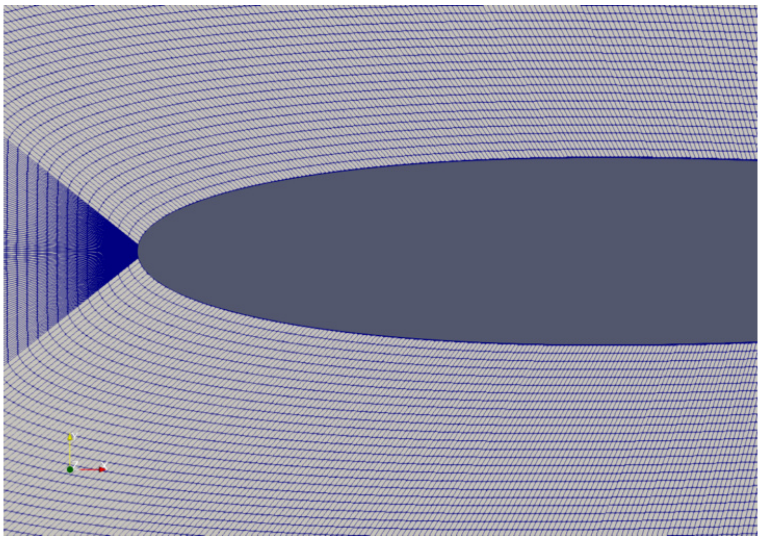
Figure 4. Demonstration of the mesh around the leading edge.

3.1.2 Numerical Results

Figure 5 presents the results evaluated for velocity fields obtained for NACA 0012 :

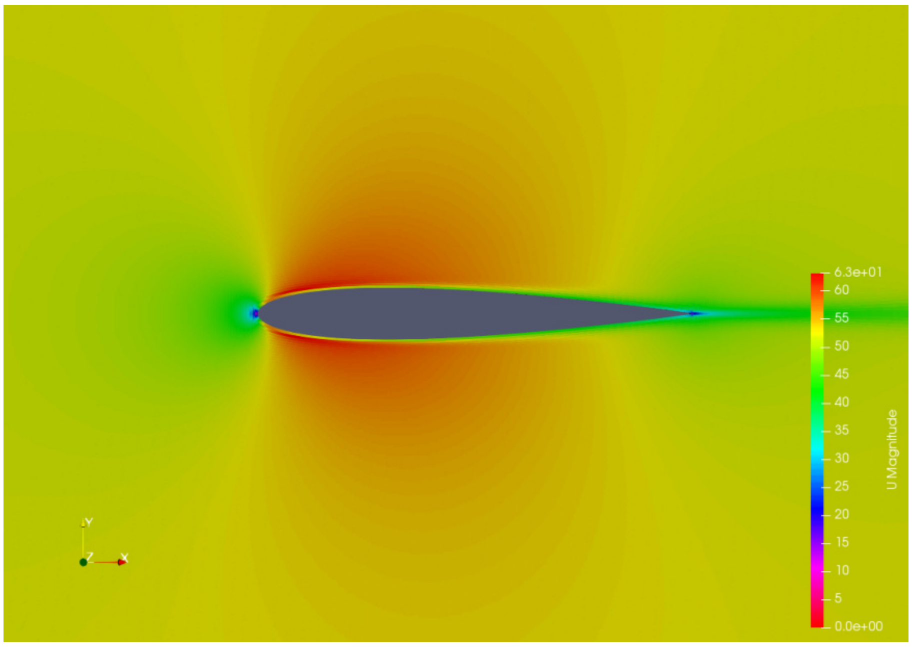

Figure 5. Velocity data from the flow field around the NACA 0012.

Figure 6 presents results for pressure fields obtained:

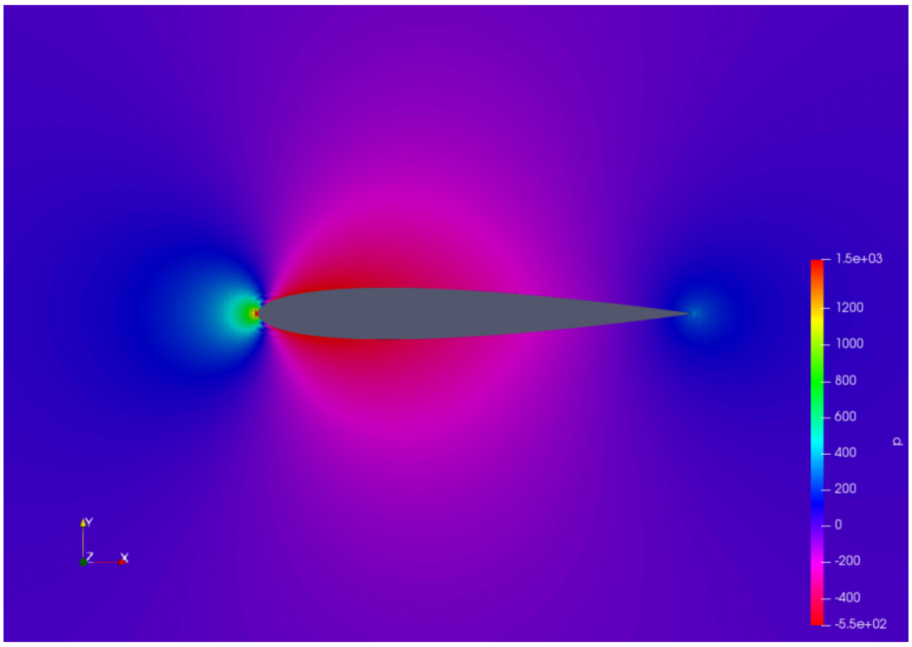

Figure 6. Pressure data from the flow field around the NACA 0012.

For the comparison with the literature, Jespersen *et al.* (2016) from NASA exercise a simulation with NACA 0012, exactly same case that was explained above.

Figure 7 presents the comparison between the experimental data provided by Jespersen *et al.* (2016) and the numerical data evaluated with the analysis for the pressure coefficient over the airfoil.

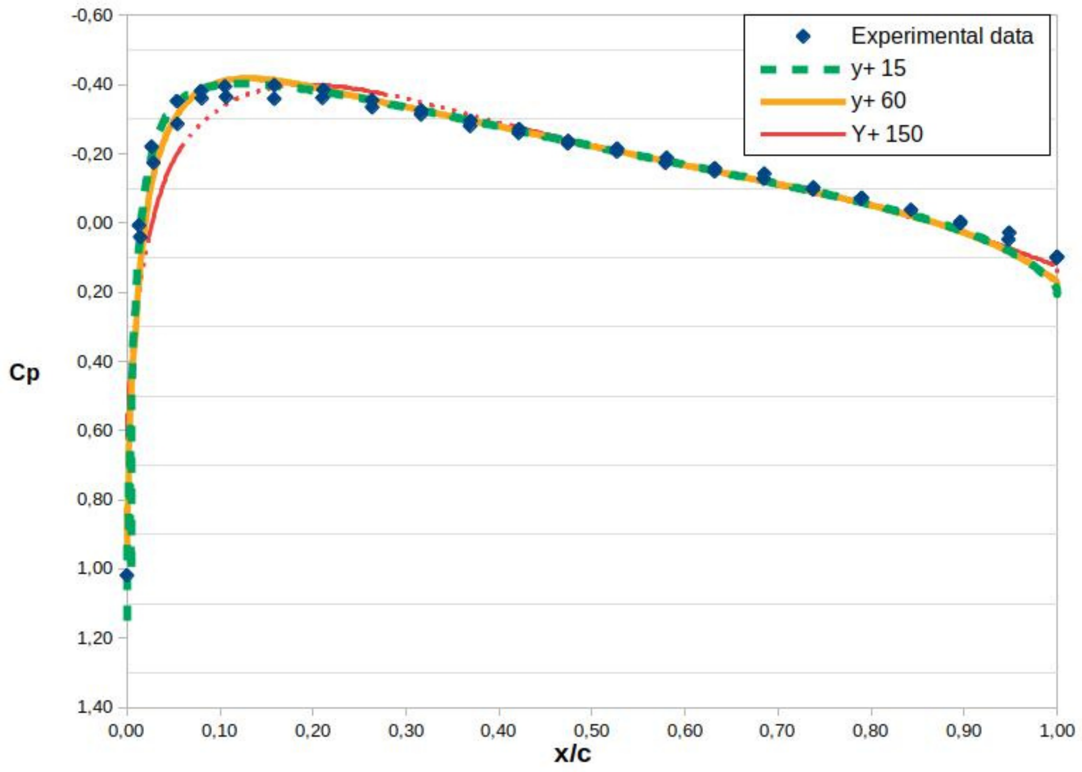

Figure 7. Comparison of the Cp evaluated with the numerical analysis with the experimental data provided by Jespersen *et al.* (2016) for NACA 0012 airfoil.

As can be seen in Figure 7 the numerical data almost fit with the experimental data, just differing slightly in the trailing edge of the airfoil.

So, following the methodology proposed by Celik *et al.* (2008), the grid convergence index (GCI) was verified for the NACA 0012 test case generated meshes. Table 2 presents the results evaluated in the GCI analysis.

Table 2. GCI analysis for NACA 0012 test case.

Number of Elements in mesh 1	7821000
Number of elements in mesh 2	1955000
Number of elements in mesh 3	782000
Refinement factor r_{21}	$1,5875$
Refinement factor r_{32}	$1,3572$
Approximate relative error e_{a21}	$1,507\%$
Approximate relative error e_{a32}	$-0,924\%$
Extrapolated relative error e_{ex21}	$4,389\%$
Extrapolated relative error e_{ex32}	$12,508\%$
Convergence index GCI_{21}	$1,8\%$
Convergence index GCI_{32}	$-15,75\%$

By comparing the results presented in table 2, and considering the methodology proposed by Celik *et al.* (2008), can be verified that the finest grid must be used in the analysis, due to GCI_{21} is less than $2,2\%$ and GCI_{32} is $15,75$.

Comparing the curves presented in figure 7 was verified that the mesh with $y+ = 15$ presented better results in comparison with $y+ = 60$ and $y+ = 150$ mesh, almost fitting with the experimental data during almost all the range o x/c, just diverging in trailing edge.

The worst results are found in the $y+ = 150$ mesh where the data diverger in trailing edge and next to the maximum Cp peak.

3.2 Analisys of NACA 0021 airfoil

Following the proposed study was studied the NACA 0021 airfoil, the results of the mesh generation, and evaluated in numerical simulation of the case are presented below.

3.2.1 Mesh

Using the points of a NACA 0021 and the method explained in the methodology, the far field has a size 20 times bigger than the airfoil chord. The mesh disposed in the following images (Figures 8 and 9) was generatedfor an $y+$ value of 60 , with approximately 2 millions hexahedral elements, since OpenFoam operates with a three dimensional mesh, even for bi-dimensional cases. Other two mesh with different values of $y+$ were generated for CGI analysis.

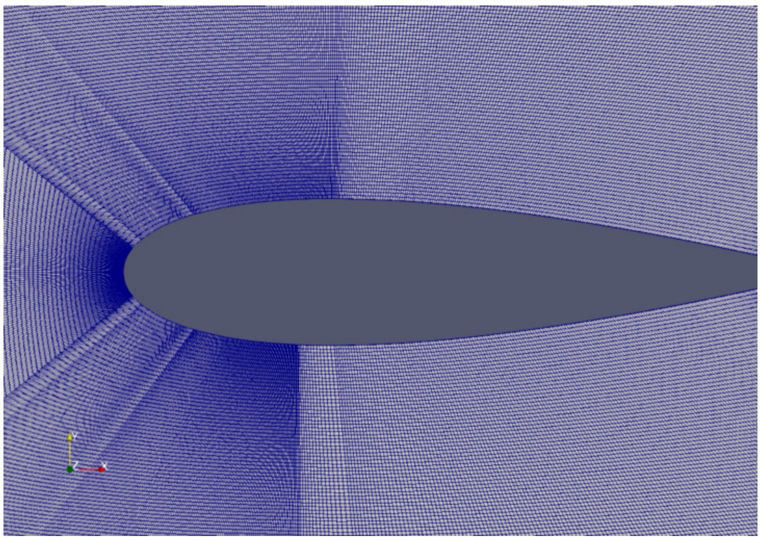

Figure 8. Demonstration of the mesh generated by Authors.

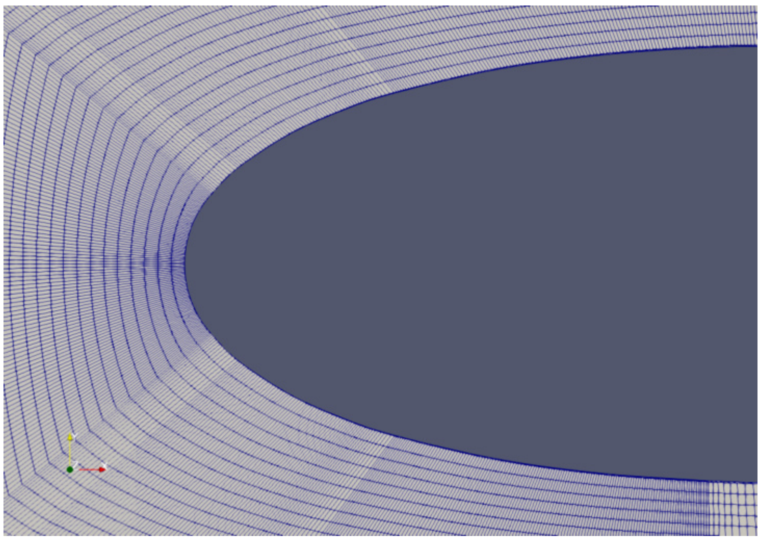

Figure 9. Demonstration of the mesh around the leading edge.

3.2.2 Numerical Results

Figure 10 presents the results evaluated for velocity fields obtained for NACA 0021 :

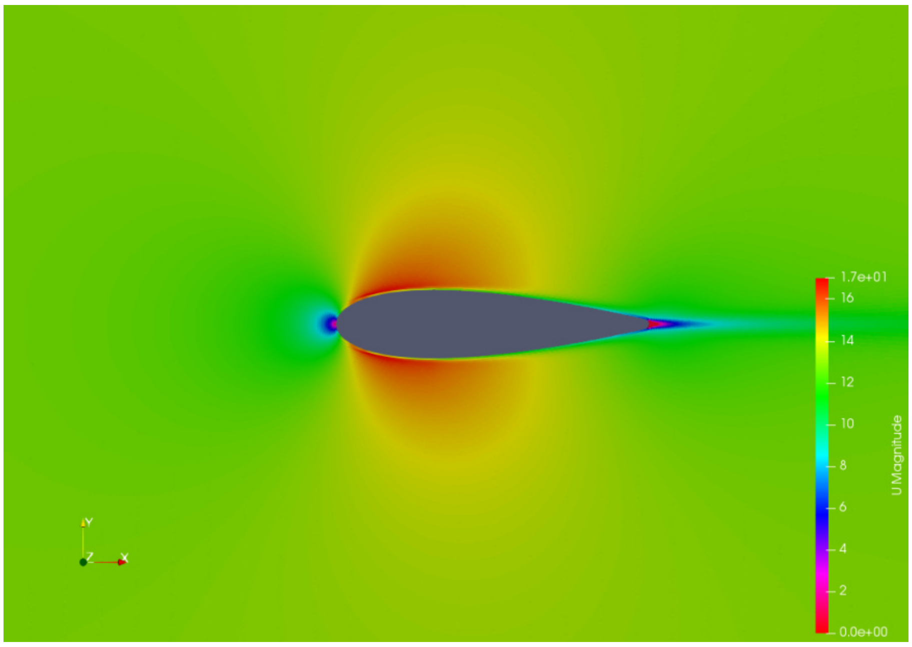

Figure 10. Velocity data from the flow field around the NACA 0021.

Figure 11 presents results for pressure fields obtained:

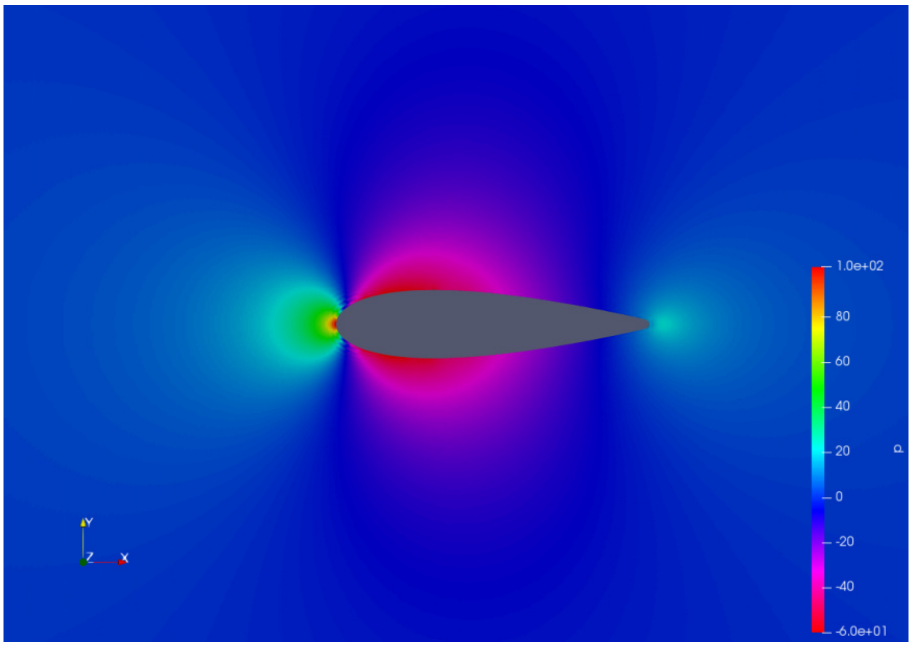

Figure 11. Pressure data from the flow field around the NACA 0021.

Figure 12 presents the Cp comparison between the experimental data provided by Wolfe and Ochs (1997) and the numerical data evaluated with the analysis over the airfoil NACA 0021.

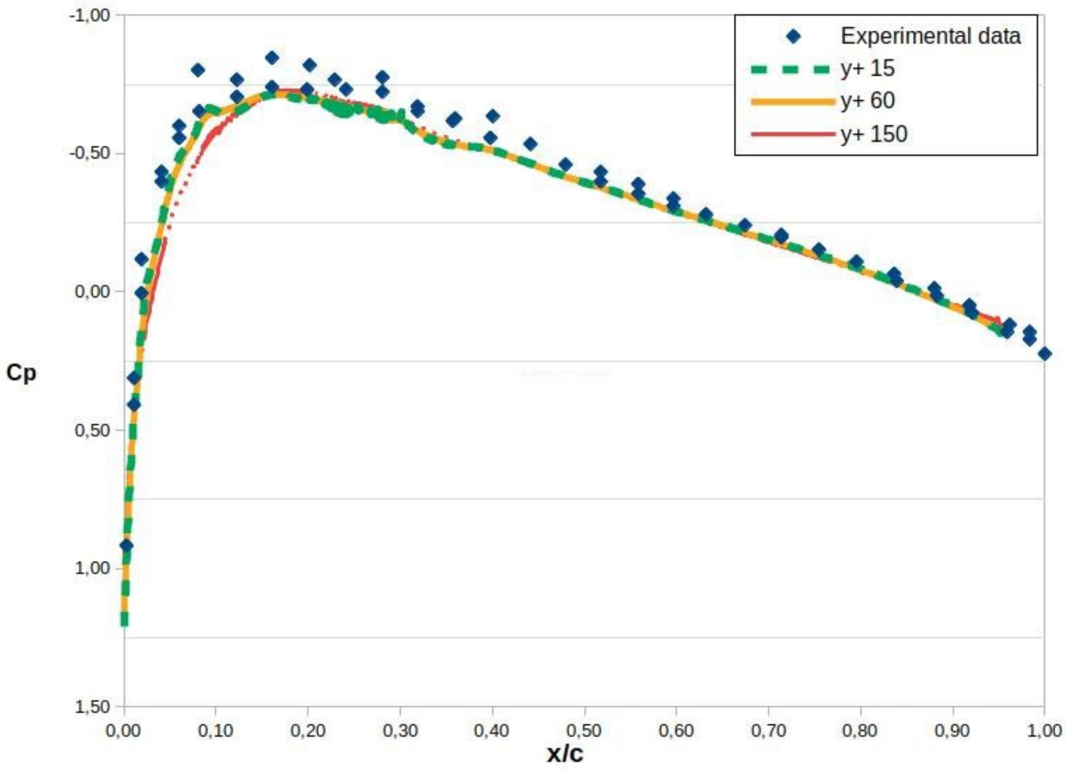

Figure 12. Comparison of the Cp evaluated with the numerical analysis with the experimental data provided by **?** for NACA 0012 airfoil.

As can be seen in Figure 12, the numerical data evaluated in the analysis keeps the same behavior of the experimental data presented by Wolfe and Ochs (1997). However, the numerical results diverge slightly from the values found by the experiments.

Another issue found in the NACA0021 case was the representation of the flow over the leading edge. In this region, the numerical data presented a significant difference from the experimental values of Cp.

Differing from the NACA0012 test case, where was found a problem in the representation of the flow over the trailing edge, the numerical analysis was capable of representing the flow over the trailing edge adequately for the NACA0021 case, fitting experimental and numerical data in this region.

Following the procedure of Celik *et al.* (2008), was verified the GCI for the NACA 0021 test case generated meshes, the results evaluated are presented in Table 3.

Table 3. GCI analysis for NACA 0012 test case.

Number of Elements in mesh 1	3886000
Number of elements in mesh 2	1943000
Number of elements in mesh 3	777000
Refinement factor r_{21}	$1,2599$
Refinement factor r_{32}	$1,3573$
Approximate relative error e_{a21}	$0,006\%$
Approximate relative error e_{a32}	$0,001\%$
Extrapolated relative error e_{ex21}	$0,000\%$
Extrapolated relative error e_{ex32}	$0,000\%$
Convergence index GCI_{21}	$0,10\%$
Convergence index GCI_{32}	$0,00\%$

Comparing the results presented in table 3 was be verified that the intermediary grid can be used in NACA0021 case analysis due to GCI_{21} is less than $0,10$.

Comparing the curves in figure 12 was verified that the meshes with $y+ = 15$ and $y+ = 60$ almost fit, only $y+ = 150$ mesh presented some difference, diverging from the other meshes next to the Cp peak.

4. CONCLUSIONS

The study showed that the results obtained using the structured mesh generated by the method proposed have approximately the same results with the literature with the two different cases. It demonstrate the versatility and quality of the mesh, and the CGI analysis indicates that the quantity of elements in the mesh does not need to be high for the same results, being the number for both cases approximately 2 million elements, reducing the cost of computational capacity to simulate the cases.

5. REFERENCES

Beaufort, P.A., Geuzaine, C. and Remacle, J.F., 2020. "Automatic surface mesh generation for discrete models – a complete and automatic pipeline based on reparametrization". *Journal of Computational Physics*, Vol. 417, p. 109575. ISSN 0021-9991. doi:https://doi.org/10.1016/j.jcp.2020.109575. URL `http://www.sciencedirect.com/science/article/pii/S0021999120303491`.

Benoit, C. and Péron, S., 2012. "Automatic structured mesh generation around two-dimensional bodies defined by polylines or polyc1 curves". *Computers & Fluids*, Vol. 61, pp. 64 – 76. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2011.09.009. Onera Scientific Day.

Celik, I., Ghia, U., Roache, P., Freitas, C., Coleman, H. and Raad, P., 2008. "Procedure for estimation and reporting of uncertainty due to discretization in cfd applications". *Journal of Fluids Engineering*, Vol. 130.

Faghih-Naini, S., Kuckuk, S., Aizinger, V., Zint, D., Grosso, R. and Köstler, H., 2020. "Quadrature-free discontinuous galerkin method with code generation features for shallow water equations on automatically generated block-structured meshes". *Advances in Water Resources*, Vol. 138, p. 103552. ISSN 0309-1708. doi:https://doi.org/10.1016/j.advwatres.2020.103552. URL `http://www.sciencedirect.com/science/article/pii/S0309170819310735`.

Jespersen, D.C., Pulliam, T.H. and Childs, M.L., 2016. "Overflow turbulence modeling resource validation results". *NAS Technical Report: NAS-2016-01*.

Liu, C. and Hu, C., 2018. "Block-based adaptive mesh refinement for fluid–structure interactions in incompressible flows". *Computer Physics Communications*, Vol. 232, pp. 104 – 123. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2018.05.015.

Lu, F., Pang, Y., Jiang, X., Sun, J., Huang, Y., Wang, Z. and Ju, J., 2018. "Automatic generation of structured multiblock boundary layer mesh for aircrafts". *Advances in Engineering Software*, Vol. 115, pp. 297 – 313. ISSN 0965-9978. doi:https://doi.org/10.1016/j.advengsoft.2017.10.003.

Lu, F., Qi, L., Jiang, X., Liu, G., Liu, Y., Chen, B., Pang, Y. and Hu, X., 2020. "Nnw-gridstar: Interactive structured mesh generation software for aircrafts". *Advances in Engineering Software*, Vol. 145, p. 102803. ISSN 0965-9978. doi:https://doi.org/10.1016/j.advengsoft.2020.102803. URL `http://www.sciencedirect.com/science/article/pii/S0965997819306891`.

Moukalled, F., Mangani, L. and Darwish, M., 2015. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab*. Springer Publishing Company, Incorporated, 1st edition. ISBN 3319168738.

OpenFoam, 2020. "Openfoam user guide". URL `https://www.openfoam.com/documentation/user-guide/`.

OpenFOAM Foundation, 2020. "Openfoam v7 homepage". URL `https://openfoam.org/version/7/`.

Pantaleão, A.V., Andrade, C.R. and Zaparoli, E.L., 2003. "A review of the delaunay mesh generation for heat transfer finite element analysis". *COBEM 2003*.

Rumsey, C., 2019. "2dn00: 2d naca 0012 airfoil validation case". <https://turbmodels.larc.nasa.gov/naca0012numerics_val.html>.

Schmidt, J., Peralta, C. and Stoevesandt, B., 2012. "Automated generation of structured meshes for wind energy applications". *Open Source CFD International Conference, London*.

Wolfe, W.P. and Ochs, S.S., 1997. "Predicting aerodynamic characteristics of typical wind turbine airfoils using cfd".

Yu, F., Zhang, H., Class, A., Xiao, J., Travis, J.R. and Jordan, T., 2019. "Winding number based automatic mesh generation algorithm for hydrogen analysis code gasflow-mpi". *International Journal of Hydrogen Energy*, Vol. 44, No. 26, pp. 14070 – 14084. ISSN 0360-3199. doi:https://doi.org/10.1016/j.ijhydene.2019.03.231. URL `http://www.sciencedirect.com/science/article/pii/S036031991931290X`.

Zhang, Y. and Jia, Y., 2018. "2d automatic body-fitted structured mesh generation using advancing extraction method". *Journal of Computational Physics*, Vol. 353, pp. 316 – 335. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2017.10.018.

6. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.